coq

By Group 4

Background

- Based on the Calculus of Inductive Constructions
  - Machine-checked proofs for software and math

Defining logical operators [edit]

The calculus of constructions has very few basic operators: the only logical operator for forming propositions is $\forall$. However, this one operator is sufficient to define all the other logical operators:

$$
\begin{array}{rcll}
A \Rightarrow B & \equiv & \forall x : A.\, B & (x \notin B) \\
A \wedge B & \equiv & \forall C : \mathbf{P}.\, (A \Rightarrow B \Rightarrow C) \Rightarrow C & \\
A \vee B & \equiv & \forall C : \mathbf{P}.\, (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C & \\
\neg A & \equiv & \forall C : \mathbf{P}.\, (A \Rightarrow C) & \\
\exists x : A.\, B & \equiv & \forall C : \mathbf{P}.\, (\forall x : A.\, (B \Rightarrow C)) \Rightarrow C &
\end{array}
$$

“If A then B” = For any assumption x of type A, B must follow

Background

- Runs on Gallina
    - Functional and strongly typed language
    - [Specification](#) language and programming language

Verifying 0 + n = n

1. Prove that for all natural numbers adding zero to n results in n
2. Start proof block
3. Assume that n is an arbitrary natural number
4. Simplify the expression
5. We have deduced that both sides of the equation are equal, the proof is verified
6. End proof block

```
Theorem zero_plus_n : forall n:nat, 0 + n = n.
Proof.
  intros n.
  simpl.
  reflexivity.
Qed.
```

Pierce, Benjamin C., Chris Casinghino, Joshua E. Schneider, Karl Crary, James P. Sterbenz, Stephanie Weirich, and Alan Jeffrey. 2013. Software Foundations. Vol. 1: Logical Foundations. Electronic textbook. https://softwarefoundations.cis.upenn.edu/. Accessed April 22, 2025.

Failing proof

3. Define recursive (fixpoint) function (ack) to take two natural numbers as the input (m n : nat)
4. Pattern match on a pair of natural numbers (if statements)
5. If m = 0, return n + 1 (_ is wildcard)
6. If m > 0 and n = 0, recursively call ack
7. If m > 0 and n > 0, call ack with an arg to call the function again (twice)

Nested recursion will fail and function will never terminate (not structural recursion)

```
Require Import Arith.

Fixpoint ack (m n : nat) : nat :=
  match m, n with
  | 0, _ => n + 1
  | S m', 0 => ack m' 1
  | S m', S n' => ack m' (ack m' n')
  end.
```

Modified: Coq Development Team. “The Ackermann Function.” *The Coq Proof Assistant, Reference Manual*, v 8.19 (Inria, 2024), § Extraction, “The Ackermann Function.” Accessed April 22, 2025. https://rocq-prover.org/doc/v8.19/refman/proofs/automatic-tactics/auto.html?highlight=ack

Why use it?

TLA+:

- Formal specifications of systems
- Model concurrent system behavior

Think: temporal model checking

THEOREM PlusZero == ∀ n ∈ Nat : n + 0 = n

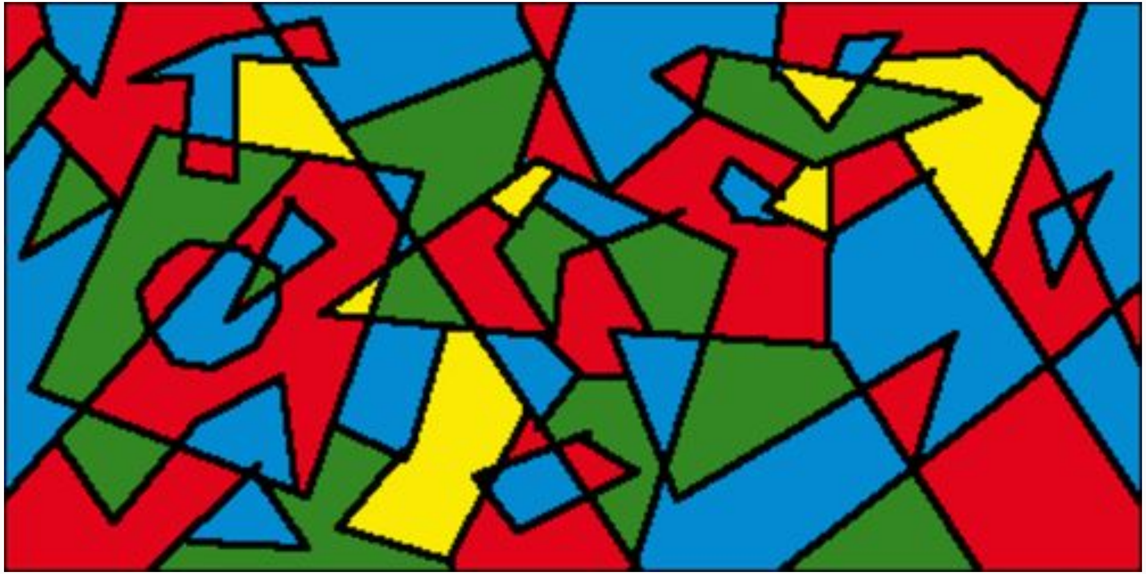
Coq:

- Formal proofs about programs and Math
- Verifying code logic and correctness

Think: machine-verified using type theory

Lemma plus_0_r (n : nat) : n + 0 = n. Proof. now rewrite Nat.add_0_r. Qed.

Why use it? - ext



- Four Color Theorem
  - Proven in 2008 with Coq that any partitioned map can be colored with 4 colors and not touch any shape with the same color (minus corners)

Gonthier, Georges. “Formal Proof—The Four Color Theorem.” *Notices of the American Mathematical Society* 55, no. 11 (2008): 1382–93.

Rating - Pros

- Expressiveness: proof automation + extraction to OCaml, Haskell (4/5)
- Well-definedness: Calculus of Inductive Constructions (5/5)
- Readability: can be intuitive to learn with those with a math/logic background (3.5/5)
- Reliability: used in CompCert C compiler (5/5)

If you value a lightweight, simple solution to logical verification, this is for you

Rating - Cons

- Very old formal method language, there are better, newer options
- More Math than Software Engineering proving

Very useful for logical conditions, not so much for temporal logic / concurrency

Additional Reading

- Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant - 2013 pdf by Adam Chlipala, MIT Press Direct
- Kaiyu Yang, Jia Deng - Learning to Prove Theorems via Interacting with Proof Assistants - CoqGym, 71K compiled human-written proofs to train a Coq proof assistant
- CoqPyt - Python framework for interacting with Coq
- CoqPilot - VSC Extension for LLM-generated Coq proofs